IoT Security Best Practices: A Critical Analysis

DAVID BARRERA, CHRISTOPHER BELLMAN, and PAUL C. VAN OORSCHOT*, Carleton University, Canada

Academic research has highlighted the failure of many Internet of Things (IoT) product manufacturers to follow accepted practices, while IoT security *best practices* have recently attracted considerable attention worldwide from industry and governments. Given current examples of security advice, confusion is evident from guidelines that conflate desired outcomes with security practices to achieve those outcomes. We explore a surprising lack of understanding, and void in the literature, on what (generically) *best practice* means, independent of identifying specific individual practices or highlighting failure to follow best practices. We consider categories of security advice, and analyze how they apply over the lifecycle of IoT devices. For concreteness in discussion, we use iterative inductive coding to code and analyze a set of 1013 IoT security best practices, recommendations, and guidelines collated from industrial, government, and academic sources. Among our findings, of all analyzed items, 68% fail to meet our definition of an (actionable) practice; 73% of all actionable advice relates to the software development lifecycle phase, highlighting the critical position of manufacturers and developers. We hope that our work provides a basis for the community to better understand best practices, identify and reach consensus on specific practices, and find ways to motivate relevant stakeholders to follow them.

CCS Concepts: • **Security and privacy → Software security engineering**; **Security requirements**.

Additional Key Words and Phrases: IoT, best practices, guidelines, inductive coding, device lifecycle

1 INTRODUCTION

Internet of Things (IoT) is commonly described as adding network connectivity to traditionally non-networked items or "things" [68]. It surrounds us with a variety of network-connected devices such as smart light bulbs, door locks, web cameras, audio speakers, thermostats, and less obvious objects like fridges, traffic lights, or sensors and controllers built into critical infrastructure systems. The importance of IoT in marketing and sales has resulted in a wide variety of devices with arguably unnecessary functionality (e.g., internet-connected toasters and toys). These devices, while offering convenience or new functionality, have acquired a reputation [6] of poor security and misconfiguration, leading to huge numbers of network-accessible devices being vulnerable. As IoT devices may be more isolated or resource-constrained (e.g., battery power, processors, memory) than their Internet of Computers (IoC—i.e., pre-IoT devices such as mobile phones, laptop/desktop computers, servers) counterparts, or lacking in software update support, their security issues are often hard to address. The cyberphysical nature of IoT—interfacing with physical world objects—also results in threats to our physical world, as well as to networks and other internet hosts [42]. This has resulted in considerable attention (e.g., [6, 19, 23, 46]) to best practices for IoT security.

The term *best practice* is commonly assumed to be intuitively understood, yet even academic work in this area (as noted below) lacks concrete definitions. We argue that this assumption is dangerous, contributes to security problems, and that at best, intuitive understandings are foggy and differ considerably across individuals. Closer inspection suggests a clear explicit definition is missing, and needed. For example, in considering Cloud Security Providers (CSPs), Huang et al. [37] refer to: "*security mechanisms that have been implemented across a large portion of the CSP industry [are thus] considered standardized into a 'best-practice'.*" Here, best practice appears to mean *widely implemented*. In their evaluation of home-based IoT devices, Alrawi et al. [6] note numerous violations of security design principles, and assert

*Authors listed in alphabetical order. This is the author's copy for personal use. Journal paper currently under submission.

Authors' address: David Barrera; Christopher Bellman; Paul C. van Oorschot, Carleton University, Ottawa, Ontario, Canada.

"*Best practices and guidelines for the IoT components are readily available*", but offer neither citations for best practices among 108 references, nor their own definition. In a recent national news article [40] on banks disclaiming liability for customer losses from e-transfer fraud, and one-sided online banking agreements, a defensive bank representative is quoted: "*We regularly review our policies and procedures to ensure they align with best practices.*" This quote appears to be not about security, but rather legal best practices in the sense of *our agreements are no worse than our competitors'*. Large collections of documents from industrial, government, and academic sources also conflate best practice with common terms such as *recommendation* and *guideline* [18]. How do best practices, good practices, and standard practices differ? Or guidelines, recommendations, and requirements? If something is not *actionable*, does it make sense to recommend it as a best practice?

In this paper, we provide what we believe is the first in-depth technical examination of intended meanings of the term *security best practice*, and the surrounding related terms noted above. For concreteness, herein our primary focus is security and privacy best practices for consumer-focused Internet of Things (IoT) devices. We argue that confusion and ambiguity result from the lack of a common understanding and precise definition of these terms, and that this confusion permeates official best practice recommendations. We support this argument by first investigating current use of terms related to best practices, and explain how their meanings differ qualitatively (Section 3). We classify these descriptive terms into three categories. We distinguish and define (actionable) security *practices* distinct from *desired security outcomes* and security *principles*. Our examination of terminology highlights ambiguity and conflation of established terms, contributing to the challenges we uncover in current IoT security advice and technical literature.

As further contributions, we offer uniform, consistent terminology, and then analyze the UK government's *Code of Practice for Consumer IoT Security* [20]. Our analysis finds it to be a surprisingly vague set of security advice (Section 3.2) despite being offered (positioned) as "practical steps" to be taken by IoT security stakeholders, featuring 13 "outcome-focused guidelines" derived from industry advice. We develop a new coding tree (Section 4) for categorizing security advice based on the terminology refined herein. Applying it to a dataset of 1013 IoT security advice items from industrial, government, and academic sources compiled by the UK government [21], we find only 32% of the 1013 items are actionable (Section 5), highlighting a gap between the expectations of entities providing advice and those intended to implement it. Our goal is not to criticise the UK group—their advice dataset simply aggregates other sources—but to demonstrate what we view as the ineffective state of IoT security advice as a whole, and to take first steps to repair this. We believe that our contributions may be of use to the broader security community, beyond IoT itself.

2 BACKGROUND AND PRELIMINARY ANALYSIS

In this section we discuss a few key areas of IoT and their role in the adoption of security best practices. These topics play a foundational role in our discussion for security advice, including when in a device's lifecycle security advice is followed, the stakeholders that have the most significant impact on the security of a device, and existing "in the wild" security advice, which we later analyze.

2.1 Lifecycle of IoT Devices

The lifecycle includes major phases a device could go through from its early design up to the time it is discarded (possibly re-used, or never used again) [32]. This plays a significant role in the discussion surrounding best practices for IoT devices, as their nature is to be long-lasting, fairly idle, and commonly link our physical environments to our digital environments. Decisions made within one part of the lifecycle may affect later phases. Once these products have left the hands of manufacturers, it becomes more challenging to properly address vulnerabilities. We believe it is

important that we understand what processes take place within each major phase, as they relate directly to the security advice that will need to be followed during them. Fig. 1 presents our model of a typical lifecycle of an IoT device based on existing work [32], and has been modified to incorporate what we believe are the most relevant milestones. Our model highlights our interpretation of the four major phases where IoT security advice is intended to be followed. Our analysis (Section 5) includes a discussion surrounding the impact of security advice followed at each lifecycle stage.

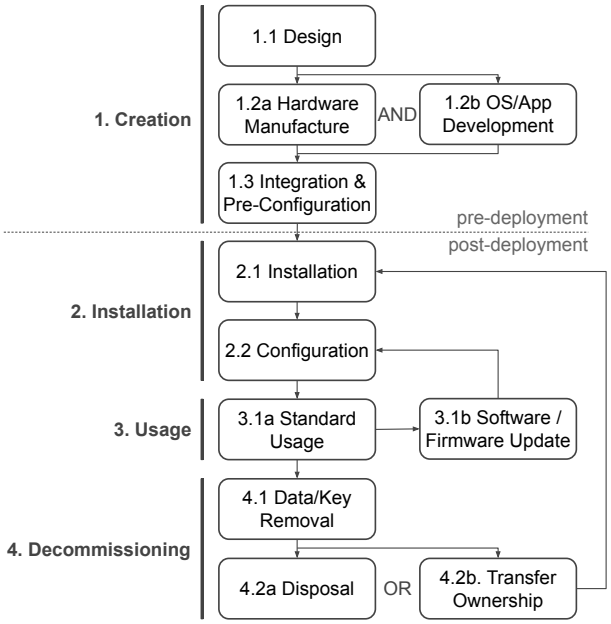


Fig. 1. IoT device lifecycle (initial design to end-of-life).

The Creation phase takes place under the authority of the manufacturer, where a product is designed, developed, and readied for use. The Creation phase happens *pre-deployment*, i.e., occurs before the product is sold to an end-user. This excludes when a user purchases the product from another user or other transfers to second users.

In the Installation phase, the user has received the product, but has not yet readied it for standard usage where the device will be used for its primary function. This phase marks the beginning of post-deployment usage. It contains *onboarding* or *bootstrapping* (often used interchangeably or meaning slightly different things, depending on who uses it) [59] including technical details such as key management, identification of devices, and establishing trust relationships; and more general user-focused configuration.

The Usage phase involves the device being used as intended (e.g., a light bulb provides light, a smart thermostat controls temperature, a home security camera provides live camera access). While appearing brief within our model (i.e., the phase only contains two events, compared to, e.g., the four of the Creation phase), this is the phase where the device is expected to spend most of its life. Software/firmware updates take place in this phase.

The Decommissioning phase is where a device ends its life with a single user/organization. The device is readied for removal from its environment (data/key removal from the device), physically removed, and leaves the end-user's ownership (either via disposal or transfer of ownership to another user). If device ownership is transferred to another user, it returns back to the Installation phase where the post-deployment ownership phases begin again.

Table 1. Assignment of UK DCMS guidelines [20] to lifecycle phases.

UK Guideline [20]	Lifecycle Phase (from Fig. 1)
UK-1 *No default passwords* [16, 19, 25]	1.3
UK-2 *Implement a vulnerability disclosure policy* [12, 25, 44]	1.1 3.1a
UK-3 *Keep software updated* [4, 12, 13]	1.1 1.2b 3.1b
UK-4 *Securely store credentials and security-sensitive data* [16, 17, 25]	1.2a 1.2b 1.3
UK-5 *Communicate securely* [12, 17, 25]	1.2b 1.3
UK-6 *Minimise exposed attack surfaces* [5, 8, 15]	1.2a 1.2b 1.3
UK-7 *Ensure software integrity* [25, 35, 63]	1.2b
UK-8 *Ensure that personal data is protected* [3, 5, 14]	1.2a 1.2b 1.3
UK-9 *Make systems resilient to outages* [12, 13, 25]	1.1 1.2b
UK-10 *Monitor system telemetry data* [15, 16, 25]	1.1 1.2b 3.1a
UK-11 *Make it easy for consumers to delete personal data* [26, 51, 61]	1.1 1.2b
UK-12 *Make installation and maintenance of devices easy* [44, 61, 63]	1.1 1.2a 1.2b
UK-13 *Validate input data* [25, 52, 61]	1.2b

2.2 Established IoT Security Advice

The UK government's Department for Digital, Culture, Media & Sport (DCMS) offers two documents regarding consumer IoT security advice. In the first, *Code of Practice for Consumer IoT Security* [20], they propose 13 guidelines (also used in Australia [9]), which are intended to provide stakeholders a set of practical security advice. Each guideline lists a brief summary title and a more detailed description of the guideline. Table 1 lists all 13 guideline titles including our assignment of where in the IoT lifecycle these guidelines could be carried out and examples of their source documents suggesting such advice. The DCMS guidelines target four stakeholders: device manufacturers, IoT service providers, mobile application developers, and retailers [20]. In the informal (unrefereed) literature, despite the lack of best practice definitions, countless documents offer advice, a subset of which are cited in Table 1.

The second DCMS document, *Mapping of IoT Security Recommendations, Guidance and Standards to the UK's Code of Practice for Consumer IoT Security* [21], collects 1052 items of IoT security advice for manufacturers of IoT products (the Version 3 dataset from [18], henceforth the "DCMS dataset/collection"), and maps each item to one of the 13 guidelines from the first document. We chose this dataset for our analysis as it represents, to our knowledge, the most comprehensive publicly available collection of IoT security advice. This advice was extracted from existing academic, industry, and government documents for manufacturers of IoT products [21]. We manually filtered the dataset to remove duplicate items (i.e., if one advice item was an exact duplicate of another, only one of the two were kept). After removing duplicates, 1013 items remained. The security advice (henceforth *items* or *advice items*) originated from 69 mostly informal documents, from 49 different organizations, and are intended to be security guidance rather than extremely detailed specification-level advice [21] (Section 3 elaborates on these terms). Some of the organizations represented in the collection includes major organizations such as the IoT Security Foundation [38], the European Union Agency for Network and Information Security (ENISA) [25], and the GSM Association (GSMA) [35]. Industrial security-focused organizations comprise the most highly-referenced sources in the collection; however, a number of government entities and businesses are included such as the US Senate [67], the US National Telecommunications and Information Administration (NTIA) [66], and Microsoft [46]. The preceding list is not meant to be exhaustive, but represents the breadth of organizations that have contributed advice to the IoT security space.

For this work, while we start with a brief analysis of the 13 guidelines for context (e.g., Table 1), we are primarily focused on an analysis of the large set of IoT security advice that the second document uses [21] (see also [18])—our main interest in the DCMS mapping of advice and the 13 guidelines is the terminology therein.

The European Telecommunications Standards Institute (ETSI) has also published a document of "baseline requirements" for IoT security [23] that appears to be an evolution of the DCMS 13 guidelines document. It includes all 13 categories as major headers (most often with slightly modified wording), but elaborates on each with a set of requirements that fit the theme of that category, much like what is done in the DCMS mapping document. For example, Provision 5.1-2 from the section titled *5.1 No universal default passwords* (comparable to the DCMS' *No default passwords* guideline header) states [23]:

> *Where pre-installed unique per device passwords are used, these shall be generated with a mechanism that reduces the risk of automated attacks against a class or type of device.*
>
> *EXAMPLE: Pre-installed passwords are sufficiently randomized.*
>
> *As a counter-example, passwords with incremental counters (such as "password1", "password2" and so on) are easily guessable. Further, using a password that is related in an obvious way to public information (sent over the air or within a network), such as MAC address or Wi-Fi SSID, can allow for password retrieval using automated means.*

This ETSI document provides advice much like the DCMS mapping document [21] and dataset [18], typically including technical details and implementation examples. We mention it here for context, as it is related to the DCMS work, but do not analyze this document further—the majority of the advice items' topics contained therein being already represented in the large DCMS dataset we analyze in Sections 4 and 5.

3 DEFINING WHAT 'BEST PRACTICE' MEANS

In this section we give a definition for *best practice*, taking into account the concepts of outcomes, actions, and *actionable* practices, and discuss related terms commonly appearing in the literature. Through this, we provide a more precise vocabulary for discussing best practices. We aim to disambiguate the wide variety of qualifying terms into meaningful categories.

3.1 Definition and Discussion

The definition of best practice seems to be taken largely for granted, as few documents that use it make an effort to define it. Of note, even RFC 1818/BCP 1 [54], the first of the IETF RFCs specifying what a Best Current Practice document is, fails to define best practice. This suggests the term (and concept of) best practice is used casually, versus scientifically—presumably everyone understands what it means well enough to not require a specific definition. This, however, leads to ambiguity, where certain uses of best practice have different meanings and connotations, while elsewhere different phrases may imply the same concept.

While focused more on human aspects of best practices, of specific note is King's discussion [41] of security best practices, where it is defined as "*practices that have proven effective when used by one or more organizations and which, therefore, promise to be effective if adapted by other organizations*". King's discussion of the definition covers a number of important concepts, including that effectiveness is based on evidence of multiple instances (implying some degree of consensus), the practice must be applicable to real situations, not theoretical; and that it may exist among a set of practices of equal quality to perform a security process [41].

In an effort to reduce ambiguity in terminology, we suggest the following informal definition: *For a given desired outcome, a best practice is a means intended to achieve that outcome, and that is considered to be better than, or at least as "good", as the best of other widely-considered means to achieve that same outcome.*[1] It is something that can be done (an action), not something that is desired to be achieved (an outcome). A community in which a best practice is developed may have their own measure for quality, as requirements may vary based on context. Further, as best practices are specific to the outcome they wish to achieve, there is generally no "silver-bullet" best practice for use across all applications—practices typically must be tailored for the context [60, Chapter 2] (e.g., a surgeon likely has different hand-washing best practices than an individual preparing a meal for themselves). Many best practices may be intended for manufacturers, but for the benefit of end-users (other stakeholders are often involved). Benefiting stakeholders may or may not be involved in a best practice's implementation. For example, a user of a product is likely not concerned with how a manufacturer implements a best practice, although end-users may rely on it for security.

One can also consider the concept of a best practice from legal, technical, and social angles. From a legal perspective, following a best practice or standard may be used as an argument to escape or limit liability, as in "following the crowd" or consensus as surely being reasonable. An example is financial institutions citing "industry best practices" to disclaim liability, per our introduction [40]. Technically, a best practice is often the best way known to technical experts or researchers, for achieving an outcome (supported by some form of consensus). Socially, best practice often implies the most common (if not necessarily best) way to do something. At one level, one might argue that each of these are similar, but at a deeper level, their semantic meanings are different uses of the same term.

3.2 Outcomes vs. Actions

We define an *outcome* as the statement of the desired end goal that a stakeholder aims to reach, and an *action* as an operation of one or more steps carried out by a person or computer, perhaps in order to achieve a desired outcome. For example, an outcome may be *have a strong password*, and an action to (partially) achieve this outcome may be *have a minimum password length of 8 characters* [34]. In practice, outcomes or goals that are vague or broad may not give stakeholders a clear idea of any concrete set of actions that would achieve the goal. A desired outcome of "strong security", for example, is nebulous, and cannot be mapped to specific actions to achieve the goal. Defining tightly-scoped outcomes or specifying an objective to withstand specific attacks allows for successful mapping to corresponding actions.

Once an action is specified or implied, the practice may be viewed as *actionable*, meaning that it can be acted upon by an advice target, and is a crucial characteristic that describes advice as unambiguous and specific about what a follower should do. We define an *actionable* practice as *a practice involving a known, unambiguous sequence of steps, whose means of execution is generally understood.* While we use *actionable practice* here to emphasize that a practice must be one that a target subject can actually do, describing a practice as actionable is redundant, as all practices are necessarily actionable by our definition (Section 3.1). An outcome cannot be a best practice via this same definition, as it does not specify a means to an end. In what follows, when we use the term *practice*, we generally mean a practice that is actionable. Likewise, if advice is positioned as a practice but is non-actionable, we consider it non-actionable advice instead of a non-actionable practice.

It follows that a recommendation specifying an outcome, but the path to which is an open research problem, cannot (and should not) be considered a practice. Requiring techniques that are experimental or unproven introduces ambiguity

[1] While one view of "best" might imply being above all known others, another is that "best" is a category that may have more than one member. It is thus reasonable to allow (by definition) that there are multiple best practices for a given desired outcome (consistent with King, above).

in how to carry out a practice and results in inconsistent execution of the intended practice (which is not, by our definition, a practice at all). We believe that it is important for the security community—whether by academic, industrial, or government efforts—to identify and agree on practices with concrete desired outcomes for use by those following the advice. Best practices that are adopted for specific use-cases will ideally both simplify, and improve the security of systems. Our heavy focus herein on (actionable) practices is because we believe, if stated clearly, they may be the most promising, directly applicable way to help pre-deployment stakeholders improve security.

Using the above definition of *actionable*, we return attention to the 13 DCMS guidelines (Table 1) and find only one of the 13 guidelines' descriptions (included in the first DCMS document [20]) fit our definition of actionable (*UK-1*: "*No default passwords*"). As a consequence, we expect that most are unlikely to be reliably implemented from this advice alone. For example, *UK-5* ("*Communicate securely*") states [20]:

> *Security-sensitive data, including any remote management and control, should be encrypted in transit, appropriate to the properties of the technology and usage. All keys should be managed securely. The use of open, peer-reviewed internet standards is strongly encouraged.*

This guideline is not actionable, as it is vague and non-specific about which actions to take to follow it, and is unfocused on a single security topic (discussed in Section 5.3). Guidelines may have implementation details inferred based on the experience of the implementer, but this does not appear to be the way these 13 guidelines are positioned. The associated DCMS mapping document [21] is intended to provide additional details and context for how the guidelines should be followed, but as we later find (Section 5), the advice found in the mapping document is largely non-actionable.

Advice authors must understand who the intended target audience is (i.e., understand their level of knowledge and limitations) in order to create suitable advice. A sequence of steps described as *generally understood* (from our definition of *actionable* above) implies that the target audience has the appropriate level of knowledge to follow the advice (independent of having sufficient resources, discussed below). Advice that is not understood or not specific enough to implement by the target audience becomes non-actionable to them (the audience), even if others may be able to follow the advice. This means wording and outcomes must be understood from both a (semantic) language perspective and, importantly, a technical perspective.

A practice does not necessarily need to specify a full sequence of low-level, specific, detailed steps; it may be acceptable to state high-level steps, provided they are still actionable. For example, a practice involving the use of AES does not necessarily require a line-by-line implementation as specified by NIST [49], but it may be enough for the practice to state a library or function to use, how it should be used, and specify any desired configuration details. To use a non-security example, a car mechanic does not need to build a car's alternator from scratch, but they are expected to be able to follow a guide to install and configure a pre-assembled one. This further highlights the importance of an appropriate target audience selection—depending on the context, specification-level details are important for, e.g., those building libraries and toolkits (to use the AES example), while others may require only the details needed to properly use the libraries provided to them. Both situations can have practices developed for them, while reaching desired outcomes and being appropriate for their respective audiences.

We separate the concepts of a practice being actionable, and an implementer having the means by which to put said practice into place. Implementers must have the resources (technical, financial, personnel) available before a practice can be implemented, but availability of resources does not affect the generic actionability of a practice by our definition. (In other words: though a practice is actionable in general, that does not guarantee that a given party themselves has the resources to adopt the practice.) A practice that has a significant cost may be ruled out as a best practice

by a recommending group, governing body, or peer community. Similarly, while still actionable by our definition, a practice that has (for example) 300 well-defined, unambiguous steps and takes 14 years to complete would likely not be considered as a best practice. Such a practice would be considered *infeasible* (i.e., a practice that remains actionable, but viewed as impractically inefficient by a party lacking the resources to carry it out). Note this is illustrated by the continuum of the actionability of practices (Fig. 2). An *Infeasible Practice* (*P3*) is actionable, but by fewer parties (suggested by its placement toward the *actionable by fewer parties* labelled end of the continuum) than a practice requiring a *Security Expert* (*P4*)—as practically speaking, high costs reduce the number of parties able to implement a practice.

3.3 Imperative and Declarative Advice vs. Actions and Outcomes

By our definition (Section 3.1), the statement of a best practice includes specifying a means to reach a desired outcome. We briefly consider now the utility of advice items that do not specify any such means or specific set of actions. As a particular case, consider an advice item that specifies an outcome whose attainment can be verified (but leaving it to an advice follower to determine a specific means). Some advice followers may still be able to attain the outcome; and auditors could verify attainment. Would such advice—called a declarative outcome, next paragraph—be equivalent to a best practice? Not by our best practice definition, which requires a specific means; an outcome and a best practice are categorically different. Nonetheless, if the means used to reach the outcome is of secondary importance to an advice giver or authority, and their primary interest is attaining the outcome, then advice items in the form of (verifiable) declarative outcomes may in practice be useful alternatives to (actionable) best practices—for advice followers who can independently determine a means to reach the outcome. Having made this observation,[2] we proceed herein to use the (Section 3.1) definition of best practice.

As supporting context bridging to related work, we note that actions and outcomes can respectively be mapped to *imperative advice* (describing specific steps or actions to reach an outcome), and *declarative advice* (describing an end result or outcome to reach, but not one specific method by which to reach it) [11, 31]. Depending on their nature, some declarative outcomes may be verifiable, e.g., through a test that yields a yes/no answer to whether the goal was reached, or a measure that can be compared to a pass-fail threshold. For example, consider the advice item: *where a device or devices are capable of having their ownership transferred to a different owner, all the previous owner's Personal Information shall be removed from the device(s) and registered services* [61]. This could be verified, for example, by checking that any memory region designated for storing user personal information has been zeroized. In contrast, a non-verifiable advice item is *use a randomly generated salt with a minimum length of 32 bits for hashing with passwords* [34]. If we assume a verifier is only presented with the fixed-length output from a hashing function (i.e., h from $h = H(p, s)$, where p is a password and s is a salt value), this practice is not verifiable, as the output (e.g., a 256-bit hash produced by the SHA3-256 function) provides no indication of the salt's length or method of generation. While we use this as an example, in practice, salts are typically stored with the hash output—if a verifier were to recover one, they would likely have access to the other.

3.4 Commonly-Used Qualifying Terms

A number of *qualifying terms* describing some practice (e.g., *common*, *good*, *best*) are often used without definition within the literature. Being widely used might suggest that readers know (and are in universal agreement on) what authors

[2]We thank an anonymous referee for raising this question.

Table 2. Summary of categories of commonly used qualifying terms related to best practices.

Category Focus		Qualifying Terms (examples)	Suggested Usage
Quality	Über	"state-of-the-art" "gold standard"	For practices that are considered superior to all others, even if not widely adopted. These terms tend to imply top-tier quality, albeit possibly at high cost or complexity.
	Best	"best current practice" "best practice"	For the practices that are *widely-considered* to be high quality (plus widely adopted, ideally).
	Good	"recommended practice" "suggested practice" "good practice"	For practices that are beneficial to implement and improve security (versus not implementing them), without implying that better practices do not exist. Here, "recommended" and "suggested" is not meant as a formal endorsement.
Commonality		"minimum expectation" "baseline practice" "accepted practice" "common practice" "standard practice"	For practices that do not necessarily imply quality, but instead reflect wide use. Alternatively, these may be de facto practices or functionality, informally recognized by experts as generally expected.
Stipulation		"regulation" "mandatory practice/requirement" "formal standard" "code of practice" "recommendation"	For practices that are endorsed (formally) or mandated in some capacity by an organization or individual. Includes practices that may be, in some way, enforced by an entity such that there implies a negative consequence should the advice not be followed.

mean when they use one of these terms. While we expect most readers would have a general intuitive understanding of the meanings of these terms, *when* they are most appropriately used in the context of providing security advice adds important nuance to their meanings. For example, IETF BCP draft *Best Current Practices for Securing Internet of Things Devices* [47] contains advice that is described in three different ways: it is a *best current practices* document (containing advice that is considered to be the best current practices) that outlines *recommendations* (suggesting the authors in some way endorse their use), but also describes them as *minimum requirements* (their use is a minimum expectation). We associate these highlighted terms with three distinct concepts that can characterize a given advice item: quality, (Section 3.5), commonality (Section 3.6), and stipulation (Section 3.7).

In an effort to both highlight existing terminology and provide a path toward more consistent use, we categorize a number of commonly-used qualifying terms by these concepts and suggest where/when each term should be used. To this end, Table 2 summarizes categories and examples of commonly-used qualifying terms, and briefly describes suggested usage. While we primarily group terms by what we view as each term's dominant goal (i.e., identifying the quality of an advice item, how commonly an advice item is used, and acknowledging a governing authority's stipulation of the advice item), an advice item can share the characteristics of multiple concepts. For example, an advice item that is considered to be a *good practice* (quality category) can also be a *standard practice* (commonality category) through wide use, or a *best practice* (quality) can be included in a formal standard (stipulation). Table 2 does not define the commonly-used qualifying terms contained therein; rather, it describes how we suggest each term (belonging to a category) should be used. For example, here our *suggested use* for *best practice* expresses its relationship to the wide consideration of high quality (while suggesting a higher quality tier exists), while our *definition* (Section 3.1) explicitly notes that best practices are better than other high quality practices with wide consideration. In what follows, we discuss these terms in greater detail.

3.5 Category 1: Quality-based Terms

Quality-based terms provide a natural basis on which to differentiate practices. Conceptually, we order *über*, *best*, and *good* practices along a quality continuum. We note that terms used to describe practices of low quality (i.e., below good) receive less attention in literature as documents promoting security advice focus more on good than bad practices. Our definition of a good practice (the lowest quality we formally categorize) implies that anything lower does not improve security.

Über Practices: The sub-category *Über* suggests practices that are in some way superior to (other) best practices, or somehow beyond what would be considered already high quality. *State-of-the-art* or *gold standard* implies something at peak technical quality, but perhaps not yet widely adopted. Consider as a practical example: in luxury cars, a heated steering wheel. While more comfortable on a cold winter day, best practice would likely be to ensure correct function and adequate steering grip to reduce the likelihood of accidents. A heating function may be the "gold standard" or "state-of-the-art" (typically at higher cost).

Best Practices: The sub-category *Best* suggests practices widely considered to be high quality, and typically, widely adopted. While practices may exist that are technically better, best practices are widely accepted within a community to be high quality.

Good Practices: The sub-category *Good* suggests practices that improve security, but are not necessarily the best practices available. They generally are not lauded for high quality per se. A good practice often either does not have wide acceptance as being the best, or is perhaps not widely done or not considered essential despite being easy and beneficial (e.g., turning the wheels to the curb when parking on a hill). Further context may prove useful for understanding their use. For example, access control to a low-value free online newspaper account may not require a best practice (per the above definition, Section 3.1); a good practice may suffice. In other words, [41]: "*sometimes the good is good enough*".

3.6 Category 2: Commonality-based Terms

Commonality-based terms, like quality terms, often include the word *practice* (e.g., *accepted practice*, *common practice*), but their unifying trait is frequency of use rather than quality.

Baseline Practice/Minimum Expectation: These terms suggest a minimum level to be reached. We assign these to the commonality category, as it is expected that the minimum acceptable level of advice is commonly followed.

Common/Standard/Accepted Practices: These terms reflect broad usage. For example, it may (unfortunately) be common to store passwords in plaintext within a database (thus being a common practice), but that is not best practice (or even a good practice).

We repeat that commonality does not necessarily imply quality. Terms in this category are less clearly ordered than in the quality category, as many items are synonyms (e.g., *common*/*standard*/*accepted*). As *baseline*/*minimum expectation* both imply a specific level to reach, these may be considered more of a priority to be followed, thus being placed higher on the continuum than the *common*/*standard*/*accepted* practices. Correlated with commonality is the *maturity* of advice, which is the length of time that advice has been, or continues to be, followed. To follow an earlier example, while not considered even a good practice, storing passwords in plaintext has become mature [53].[3] Ideally, a best practice would

[3]We do not consider storing passwords in plaintext "advice", as it is likely there is no advice suggesting to do this other than for very specific and niche reasons; rather, the existence of plaintext passwords is the *absence* of acted-upon good advice.

be mature, and widely considered to be high quality (Section 3.5), but the quality of advice is not a determining factor of its maturity (e.g., DES is a mature cipher, but no longer best-practice). Similarly, security design principles are a known set of guiding rules which aim to improve security [58]. These principles are generally based on experience, suggesting their maturity. Security design principles would also generally be expected (by experts) to be followed, and they are complementary to the existing categories, but are omitted from Table 2, and discussed further in Section 4.3.

3.7 Category 3: Stipulation-based Terms

While the above discusses quality and commonality, some terms related to best practices have more to do with the endorsement by an authority, the authority's jurisdiction, and whether the advice is mandatory (i.e., a firm requirement). Note that the entity creating advice is not necessarily the authority mandating its use. The *stipulation* category notes qualifying terms describing advice that is mandated or endorsed by an entity in some way. These too can be ordered along a continuum. On the strict end, the terms that imply a negative consequence for not following the advice (e.g., mandatory practice, requirement, regulation). On the looser end, the terms that are stipulated, but not necessarily enforced (e.g., guideline/guidance, recommendation). As with a best practice, stipulations should, in our view, ideally be accompanied by an explanation of the intended outcome.

Regulation: We use *regulation* to be a directive from an authority stating specific advice that must be followed to be allowed to operate within a *jurisdiction* (referring to the legal or authoritative domain, or the context of the deployment environment or use cases, e.g., home IoT may require different practices than IoT devices for government; physical locations, e.g., to meet certain requirements to be allowed to be sold in a country; or scope of technology, e.g., certain practices may be more appropriate for IoT devices rather than desktop computers).

Mandatory Practices/Requirements: Hereafter just "*requirements*", these do not necessarily imply the quality of a given practice, but rather that it is stipulated by some governing body or regulation, suggesting official endorsement. These may be considered "enough" for some purposes (e.g., *enough to not be sued* or *enough to pass inspection*). Practices across a range of qualities may be requirements depending on the governing body or motivation, although a practice that has proven to be of a high quality is more likely to become a requirement or part of a standard.

Formal Standard: We take *formal standard* to mean a formally documented (adopted or endorsed by some authority) set of requirements. This implies not necessarily quality, but an established set of practices. The purpose of a standard may be interoperability—e.g., standards for the gauge of rail tracks, or gauge of screws or pipes. In this context, formal standard differs from *standard practice* (i.e., common practice, above), related to frequency of use, e.g., it is *standard* to eat at 12 noon. Standards are typically sufficiently detailed such that conformance or compliance is easily judged by, e.g., an auditor, or interoperability tests.

Code of Practice: We take *code of practice* to mean a set of advice designed to help those following the advice to reach a specific level of compliance (whether it is a formal standard, a regulation, or a requirement within the relevant jurisdiction). Codes of practice are often stipulated (e.g., within an organization, an industry), but do not imply the same level of consequence as the stricter terms (formal standard, mandatory practices/requirements). Note that a code of practice is distinct from formal regulations such as an "electrical code" or "building code".

Recommendation: A *recommendation* is an endorsement of, e.g., a practice by an individual or organization as their suggested way to do something. Recommendations (depending on the recommending entity) may be subject to bias or

be self-serving, and do not necessarily reflect expertise or universal consensus. Some recommendations, depending on their sources, are, in essence, requirements. Recommendations commonly suggest following a standard [21]. Similarly, a *guideline* or *guidance* is often given to promote a suggested way *to achieve a goal*. This may be used in the spirit of a recommendation—offered as help, versus imposing rules. We typically guide someone to something, suggesting a target, goal, or desired outcome.

As final thoughts on best practices and related terminology, we find an inconsistency in both the use of common terms (discussed throughout Section 3) and the situations in which they are used, leading to issues where the same terms are used with different meanings in different situations. Establishing consistency in use among the three concepts we delineate herein (quality, commonality, stipulation) provides a foundation to discuss and further analyze the advice provided and used within the security community.

4 EMPIRICAL STUDY: METHODOLOGY

Our methodology for the analysis of existing IoT security advice is based on iterative inductive coding [45, 64], where our goal is to categorize (*code*) 1013 IoT security advice items (as opposed to the 13 guidelines described in Section 2.2). Our coding categories rely on Section 3's terminology, with refinements as explained in subsections below.

4.1 Establishing Analysis Tools

To begin development of a codebook for inductive coding, one author first reviewed the dataset (Section 2.2) to extract rough topics for codes. Discussion and preliminary test codings based on the extracted topics by two authors resulted in the following coarse codebook (set of codes): *Practice*, *Incompletely specified practice*, *Outcome*, *Security design principle*, *Too vague to tell*, *Out of scope*.

After establishing this early codebook (with associated definitions), test sets consisting of ten new, mechanically selected items from the dataset (to test across topics within the dataset; e.g., item numbers 100, 200, [...], 1000) were coded to determine agreement between authors. This consisted of each author (a *coder*) reading, interpreting, and assigning each item in the set to a code (informally called a *tag*). This process was done a second time on a distinct set of 10 items. This refined the codebook by creating new codes for items that did not fit well into existing codes. Assigning an item directly to a code was subjective, resulting in low inter-coder agreement of between 30–40% for the first two trial tests.

To reduce subjectivity, we built a coding tree to more objectively guide coders towards codes based on *yes*/*no* questions (Fig. 5). Each question aimed to clearly distinguish codes, progressively allowing coders to disqualify branches for an advice item and move toward the leaf nodes (containing the resulting codes). This method was iteratively improved through an additional five test coding sets. Improvement was measured based on inter-coder agreement after modification (i.e., addition/modification of codes and/or questions) of the tree. Over several iterations, coders discussed the results to resolve ambiguities and gaps in definitions or questions in the coding tree, refining questions difficult to answer and coming to an agreement on the codes [36]. As a result, the coding tree evolved aiming to allow potential coders to more easily and accurately code items, improving inter-coder agreement through its concise decision path. To code items that required more reflection, coders consulted a further detailed annotation (see Appendix A).

Many advice items proposed as practices were explicit about technique or technical method to address security, but lacked actionable detail (e.g., Item #50 "*endpoints must always use standard cryptographic algorithms*" [63]). This led us to recognize the continuum of Fig. 2. On its left side, practices are much less widely actionable (a general practice/policy, an incompletely specified practice). These often specify vague technical directions to take or methods

to use ("*standard cryptographic algorithms*" from the above example), but do not specify explicit actions to allow for its successful implementation. On the right side are practices that even end-users would be able to carry-out, and suggests that if an end-user would be able to carry out the practice, so would a more experienced implementer. Moving left requires more in-depth knowledge and experience to understand (or infer a direction from) an item, and implementation details become ambiguous or unclear (even to a security expert). Coders did not use this continuum for coding, but we see this continuum as a visual representation of where each type of practice may exist in relation to each other in support of the coding tree.

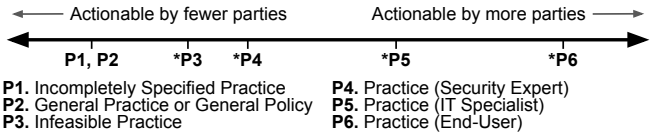


Fig. 2. Practice categories: Actionability continuum. Terms defined in Fig. 4. An asterisk (*) indicates practice categories defined as actionable. (Sections 3.2 and 5.2 discuss *P3*.)

The binary decisions made by the coders (through use of the coding tree) resulted in codes, so indirect placement onto the continuum is a side-effect. In contrast, where to directly place an advice item on the continuum (*P1–P6* in Fig. 2) may be less clear or some point between two codes. For example, Item #90 "*communications protocols should be latest versions with no publicly known vulnerabilities and/or appropriate for the product*" [61] could be coded as either *P4* or *P5* depending on assumed knowledge of the implementer. By convention, we considered agreement among coders if both coded the item as a practice, and either of their codes were within plus-or-minus one ("±1") code distance away on the continuum (Fig. 2). For example, if one coder coded an item as an *Infeasible Practice* (*P3*) and the other coded it as a *Specific Practice—Security Expert* (*P4*, one position to the right of *P3*), this would be considered an agreement as they are close enough on the continuum to have the difference be reasonably accounted for by subjectivity. Fig. 3 relates practices in this continuum (Fig. 2) with other concepts, and is discussed in Section 4.3.

To further address subjectivity in use of the coding tree, a second code was made available to the coders. If a coder reached a question that, based on their interpretation of the item, could be viewed as both a *yes* or *no*, both the first and second code slots could be used to follow each branch to a leaf node. This allowed coders to code an item while following the coding tree in situations where coders could make an argument for both answers. For example, Item #977 "*The RTOS makes use of secure storage to protect sensitive application data and secrets and additionally binds the data to a specific device instance*" [55] could be viewed as an *Incompletely Specified Practice* (*P1*) if there is a question about exactly what *sensitive application data and secrets* are, or a *Security Expert* (*P4*) practice if one assumes that a security expert could reasonably determine what these are. As such, neither code takes priority over the other in the analysis, and both are counted in the results (e.g., Fig. 6, Fig. 7). For calculating agreement, if coders agreed on at least one code, it was counted as an agreement for that item. In total, 16% of items used two codes.

A final trial coding was done with a set of 20 items. Based on the ±1 rule and primary/secondary codes, the mean agreement rate between the three coders was 73% (between Coder 1 and Coder 2, 80% agreement, Cohen's Kappa [45] $\kappa = 0.84$; C1 and C3, 75%, $\kappa = 0.66$; C2 and C3, 65%, $\kappa = 0.58$). The final codebook is in Fig. 4. After this trial, a detailed technical analysis and full coding of the 1013 item dataset was done by one coder using the coding tree (Fig. 5). Different from typical inductive coding methodologies where a code is manually assigned to an item by a coder, when we say an advice item is "coded" by a coder, we mean that they used the coding tree and the tree assigned the resulting code to an

item. The coding interface displayed the coding tree (Fig. 5), code definitions (Fig. 4), detailed annotations for each question (App. A), and two drop-down boxes where coders could input the codes they reached through use of the coding tree. It is preferred that coders follow the coding tree down to the leaf nodes and then enter the code they reached into the drop-down boxes, but there were no restrictions to stop coders from immediately selecting a code based on their first reading of the advice item instead of following the tree.[4] The decision to use a single coder was based on all authors agreeing on the final set of codes and questions in the coding tree (in agreement with the methodology of, e.g., Huaman et al. [36]), level of agreement during test codings, and the work effort required to manually code (via the coding tree) 1013 items. Using a single coder is noted as a limitation of this work. Supplemental work (to be reported separately) will explore a second full coding of the 1013 items by at least one additional coder and pursue detailed explanations of any major deviations found.

We note that many of the terms discussed in Section 3 are not represented in our codes. Where no context is provided surrounding an individual advice item and we are only presented with the text of the advice (as is the case with the advice we analyze herein), it is difficult to know if an item belongs to any of the three major categories from Section 3 (i.e., quality, commonality, stipulation). For example, to know the quality of an advice item requires knowledge of how a community rates a practice, to know its commonality among practitioners requires knowledge of how frequently that advice is used, and to know if it is stipulated requires knowledge of how that practice is mandated in the real-world. As such, terms like *best practice*, *common practice*, or *regulation* are not used in our codes (Fig. 4), as we cannot determine which category, if any, was intended by the advice giver without additional context. We have instead developed codes that can be applied to advice items without requiring contextual information.

As the advice in the dataset is grouped by the 13 guidelines in the DCMS mapping document [21], all advice items in the coding of the full 1013 item set were randomly ordered to avoid bias from reading similar advice in repetition.

4.2 Advice Categorization by Lifecycle Phase

Independent of the inductive coding of Section 4.1, the coder assigned each actionable item (*P3–P6*) with the phase in the IoT lifecycle (Fig. 1) where the item could be carried out, independent of the codes defined and used in the coding tree. This was done by determining which stakeholder would be in a position to carry each item out, and matching where their work occurs in the lifecycle. This determination was based on which stakeholders *could reasonably* implement a practice (within reason—an end-user given an API would not be likely to implement a best practice or fix a vulnerability), not necessarily the single stakeholder in the *best/most effective position* to implement it, thus allowing for items to be associated with multiple phases. For example, Item #191 [35]:

[4]While coding reported here generally avoided use of such short-cut coding, this check was not built into the coding tree tool's user interface. In retrospect, in a preferred implementation, the coding tree tool's user interface would either force coders to select *yes*/*no* answers until a code is automatically assigned to an advice item, or allow the short-cutting by assigning a code directly to advice, but with the use of this short-cut being recorded.

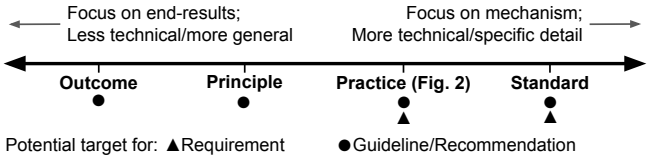


Fig. 3. Relationship between terms based on focus of advice's intent (end results vs. mechanism).

P1. Incompletely Specified Practice: Advice that suggests a technical direction of a practice (e.g., a technical method/technique, software tool, specific rule), but lacks clear indication of any steps to be taken, and fails to meet our definition of actionable.

P2. General Practice or General Policy: Advice that is not explicit about any techniques or tools, but is considered a general approach to improving security. This may also be policy-related advice. These are not considered actionable (despite being labeled as a practice) due to their general, less specific nature.

P3. Infeasible Practice: A practice, but would require an unreasonable amount of resources (e.g., time, financial, human) to implement, or cost vastly more to implement than would be gained from its implementation.

P4. Specific Practice—Security Expert: A practice requiring an expert in security to implement. These may require in-depth knowledge and experience of security topics, and often rely on the implementer to infer steps that are not clearly defined in the advice.

P5. Specific Practice—IT Specialist: A practice that IT specialists (dedicated IT and development employees) developing or maintaining a product would be able to implement. These practices do not require the implementer to be a *security* expert, but assumes a basic knowledge of computer security such as that gained through coursework in formal or informal education.

P6. Specific Practice—End-User: A practice an end-user would be able to implement. These are actionable by that audience, and typically done by the user via direct interaction with the device, using a mobile app, or cloud service.

N1. Security Principle: Advice that suggests a generic (as in applying to many situations) rule to follow that has shown through experience to improve security outcomes or reduce security exposures. See discussion in Section 4.3.

N2. Security Design Principle: Advice that suggests a *Security Principle*, but specifically for the Design phase of the lifecycle.

O1a/b. Desired Outcome: Advice that suggests a generic, high-level end goal that a stakeholder would like to reach (as opposed to a means by which to reach a goal).

M1. Not Useful (too vague/unclear or multiple items): Advice that does not make sense from a language perspective (e.g., not full sentence, unclear grammar), or is not focused on a specific task/action to complete.

M2. Beyond Scope of Security: Advice that is not clearly an item that would be implemented for the benefit of security.

Fig. 4. Codes and descriptions for coding tree of Fig. 5.

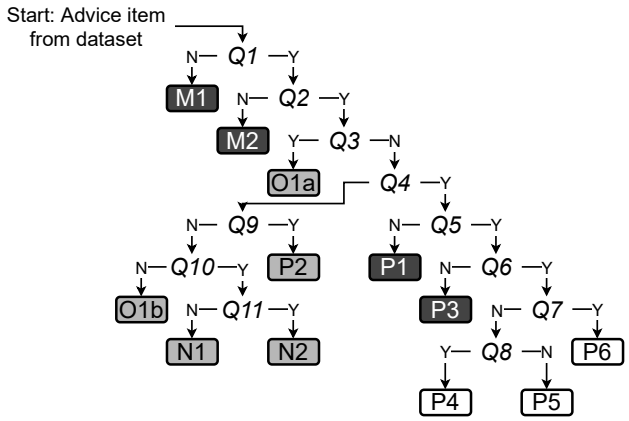
> *When a product is being developed it is often enabled with debugging and testing technologies to facilitate the engineering process. This is entirely normal. However, when a device is ready for production deployment, these technologies should be stripped from the production environment prior to the definition of the Approved Configuration.*

This item could either be done in the OS/App Development phase (1.2b) where code is stripped from software before completion, or during the Integration & Pre-Configuration (1.3) phase where features may be disabled or left out of device integration. We note that only practices (being implicitly actionable) were considered for this categorization, as without actionability, it is difficult to determine what steps would need to be taken and when (in the lifecycle) they would be followed.

While many items included in the advice statement itself, an indication where they would be carried out in the lifecycle (e.g., do not hard-code secret access codes for testing/debugging in software [39]), others required subjective judgement for placement (e.g., "*keep software updated*" [22] could be targeting the Creation phase or Usage phase depending on which stakeholder it is suggesting should maintain software). If the item did not have an obvious or implied associated lifecycle phase, it was categorized as *Unclear* (see Fig. 7).

4.3 Relationship to Security Principles

We observed that many security advice items were rephrasings of established security principles. In our context of computer security, we define a *principle* to be a generic (applying to many situations) rule shown through experience to



Q1. Is the item conveyed in unambiguous language, and relatively focused?
Q2. Is it arguably helpful for security?
Q3. Is it focused more on a desired outcome than how to achieve it?
Q4. Does it suggest a security technique, mechanism, software tool, or specific rule?
Q5. Does it describe or imply steps or explicit actions to take?
Q6. Is it viable to accomplish with reasonable resources?
Q7. Is it intended that the end-user carry this item out?
Q8. Is it intended that a security expert carry this item out?
Q9. Is it a general policy, general practice, or general procedure?
Q10. Is it a broad approach or security property?
Q11. Does it relate to a principle in the design?

Fig. 5. Decision tree for classifying advice items. Leaf node codes explained in Fig. 4. Black shading represents codes we consider not beneficial to include in advice (for lack of actionability or feasibility); white are desirable actionable practice codes (excluding infeasible practices); grey are codes considered useful for context, but not actionable like codes in white boxes (P4, P5, P6).

improve security outcomes or reduce exposures, and a *design principle* to be a subset specifically guiding the *design* of a system. Other subsets may relate to other lifecycle phases (Section 2.1). For context, note Saltzer and Schroeder [58] define eight "*[...] useful principles that can guide the design and contribute to an implementation without security flaws*". NIST [62] notes "*the primary focus of these principles is the implementation of technical controls*", suggesting that security principles are appropriate targets to be implemented via practices. In the coding tree, both security principles and security design principles are leaf nodes.

To relate security principles with practices, outcomes, and other terms, we have provided the continuum of Fig. 3. On one extreme (left) are concepts more focused on end-result (outcomes), and on the other are the most actionable items that focus on mechanisms to reach outcomes, often specified in fine detail (standards). As Fig. 3 indicates, items on this continuum may serve as a guideline or requirement (Table 2). Ideally, what is imposed as requirements by governing bodies should be practices or standards (versus principles or outcomes), as requirements should be actionable so those subject to the requirement have a clear understanding of how to follow it (see Section 3.2 for *actionability*). This is represented on this continuum by the labeling of *Practice* and *Standard* as potential targets for requirements (denoted by the triangle).

4.4 Actual Use of Methodology

We envision the methodology described in Section 4 to be used primarily in two ways. The first is for measuring the actionability of existing advice as a means to establish a general view of the current state of IoT security advice, and to determine where advice fails to meet the needs of security practitioners. This is the primary focus of the coding exercise described in Section 4, and analyzed in Section 5. The methodology can be used in a second way for the analysis of new IoT security advice, as a tool to assist advice authors in creating actionable advice. If advice authors themselves use the coding tree on their own advice items, they can differentiate actionable from non-actionable advice (among other more fine-grained types of advice). After using the coding tree, security advice that results in an undesirable code[5] can then be revisited by advice authors to reword and clarify the explanation of the advice. Once an advice item is reworded, the questions in the coding tree may be answered differently, giving advice authors feedback about whether their changes have had a positive impact on the actionability of their advice, or if it follows a path down the tree to a more desirable code. If the coding tree outputs an undesirable code (e.g., a non-actionable code), those giving the advice may be able to see (from the coding tree) at which question the advice diverged from a path to a desired code. For example: "*encrypt stored passwords*" gives vague advice and is ambiguous on how to achieve encryption. Question 5 (from the coding tree, Fig. 5) sends this advice to *Incompletely Specified Practice* (*P1*), as it lacks actions (explicit or implicit) to take. The advice could be reworded to specify what form of encryption advice authors suggest, and provide explicit references to how that encryption would be implemented, thus now passing *Q5* as actionable.

5 EMPIRICAL ANALYSIS AND DISCUSSION

In this section we discuss our analysis of the 1013-item IoT security advice dataset. Again for context, this is the same set of security advice that the DCMS used to create their 13 guidelines [20]. We coded the items in this collection using the methodology of Section 4. The primary goal of matching each item in the dataset to codes (and associated definitions) is to provide a general sense of how well existing advice lists and literature have specified practices (as opposed to advice *positioned* as practices, but failing to be actionable, as required by our definition). Identifying where practices are carried out throughout the IoT lifecycle allows us to see which stakeholders have the greatest number of items to address regarding overall device security.

5.1 Results of Coding

Fig. 6 summarizes the number of advice items assigned to each code. For distinguishing actionable vs. non-actionable advice (bottom of figure), if an item had two codes (primary and secondary) and either was an actionable code (i.e., *P3–P6*), the item was considered actionable instead of non-actionable because an argument could be made for its actionability. For example, if an item was coded both as an *Incompletely Specified Practice* (*P1*) and *Specific Practice—Security Expert* (*P4*), the item is declared actionable as *P4* is considered actionable. As such, the sum of actionable and non-actionable items in Fig. 6 adds to 1013.

The coding process allowed a coder to designate whether an item was specific to IoT. None of the items in the dataset were highlighted in this way. This suggests our methodology is broadly applicable. Following this finding, we

[5] We view the creation of actionable practices as the objective for advice authors; however, creating practices might not be the intention of all advice authors—some may instead intentionally craft non-actionable guidance in the form of Outcomes (*O1a/b*), General Practices or General Policies (*P2*), or Security Principles (*N1*/*N2*)—see Fig. 4 for descriptions of these forms of advice. Advice authors may use the coding tree to steer their advice toward non-actionable codes, but we recommend it is used in the pursuit of actionable codes (*P3–P6*).

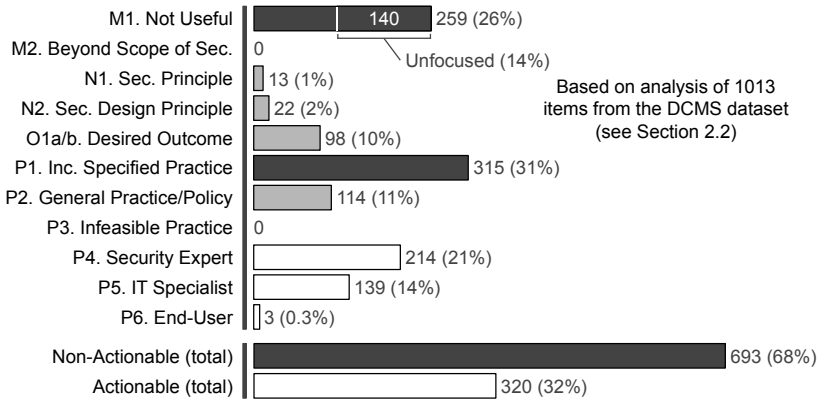


Fig. 6. Main summary of data coding. Sum exceeds 1013 as items may be assigned two codes (primary/secondary—recall Section 4.1). Shading intensity follows the same scheme as Fig. 5.

intentionally left the resulting coding tree generic (non IoT-specific). While herein, we used the coding tree to explore IoT security, its design and resulting structure apply more widely to analysis of security advice in general.

5.2 Proportion of Non-Actionable Advice

Our analysis shows that organizations—often highly credible ones—are producing recommendations for manufacturers that are not, by our definitions, actionable, thus at greater risk of being improperly implemented. We believe that this low level of actionability is a signal that the security community must significantly improve how we capture and state best practices if manufacturers are expected to follow recommendations.

The methodology used declares any code after the *yes* branch of Question 5 in the coding tree (*P3–P6*, see Fig. 5) to meet the definition of actionable (Section 3.2). In total, 32% (320/1013) of items were found to be actionable (at least one actionable code as explained in Section 4.1). 21% (214/1013) of items were coded as *Specific Practice—Security Expert* (*P4*), 14% (139/1013) as *Specific Practice—IT Specialist* (*P5*), and <1% (3/1013) as *Specific Practice—End-User* (*P6*)—see Fig. 6. The *Infeasible Practice* (*P3*) code also remained unused. This was encouraging, suggesting that advice providers have an understanding of what sorts of practices are within reason for their target audience. Similarly, the code *Beyond the Scope of Security* (*M2*) was also unused; however, this is arguably due to source documents being at the very least sufficient to target computer security.

As Fig. 6 notes, 68% of the advice was found to be non-actionable. We would expect that this significant majority of advice will either be poorly implemented despite the advice takers' best effort to understand what is asked of them, or not implemented at all, defeating the purpose of security advice. This is not to say that non-actionable advice is useless—outcomes, principles, and general practices (Fig. 4) still specify a desired end-result or generic goals to reach. Actionability may not be essential in all use cases (Footnote 5); however, our underlying assumption is that advice givers (for the documents under discussion) intend to be offering advice in the form of best practices. We argue that actionability should be considered a high (if not the highest) priority among the characteristics of security advice (see Section 3.2).

5.3 'Not Useful' Advice

During the iterative development of our coding tree, we found many advice items in the dataset (described in Section 2.2) tended to not really be things to do, but descriptions of security techniques (e.g., a hardware security module, public-key encryption) or threats to a system (e.g., unused, accessible physical ports), and offered no suggestion for something that should be implemented. Item #387 [63] provides an example of this:

> *Network firewalls are message-oriented filtering gateways used extensively to segment IIoT [industrial IoT] systems. Most firewalls are Layer 2, 3 or 4 IP routers/message forwarders with sophisticated message filters. Firewalls may be deployed as either physical or virtual network devices. A firewall's filtering function examines every message received by the firewall. If the filter determines that the message agrees with the firewall's configured traffic policy, the message is passed to the firewall's router component to be forwarded.*

One could make the argument that the description of a technique implies that the advice giver wants it to be implemented, but it reads drastically different than more typical "do this" security advice and tends to not include actionable detail. As such, we consider these types to be not useful for a stakeholder to implement. 26% (259/1013) of items were coded as *Not Useful* (*M1*)—see Fig. 6.

Similarly, individual advice is commonly given in rapid succession within a single piece of advice. As a sub-category of the *Not Useful* (*M1*) code, we added an *Unfocused* supplementary code for coders to use when they find multiple sub-advice items within one item (represented as a sub-bar of the *Not Useful* code in Fig. 6). For example, Item #84 [12] highlights this:

> *IoT Devices Should Follow Security & Cryptography Best Practices. (1) BITAG recommends that IoT device manufacturers secure communications using Transport Layer Security (TLS) or Lightweight Cryptography (LWC). Some devices can perform symmetric key encryption in near-real time. In addition, Lightweight Cryptography (LWC) provides additional options for securing traffic to and from resource constrained devices. (2) If devices rely on a public key infrastructure (PKI), then an authorized entity must be able to revoke certificates when they become compromised, as web browsers and PC operating systems do. Cloud services can strengthen the integrity of certificates issued by certificate authorities through, for example, participating in Certificate Transparency. (3) Finally, manufacturers should take care to avoid encryption methods, protocols, and key sizes with known weaknesses. (4) Vendors who rely on cloud-hosted support for IoT devices should configure their servers to follow best practices, such as configuring the TLS implementation to only accept the latest TLS protocol versions.*

This advice jumps across four topics (we inserted the numbers for exposition): (1) the use of TLS or lightweight cryptography, (2) certificate revocation, (3) avoiding weak or vulnerable key sizes, (4) then avoiding outdated TLS versions. Trying to successfully code advice such as this was a challenge, as different parts of the advice could be coded differently. We found that advice items with a longer word length were unfocused in this way. In total, we found 54% (140/259) of *Not Useful* (*M1*) codes, or 13.8% (140/1013) of all items were coded as *Unfocused* because they contained multiple distinct topics within the advice text (similar to the above example). Had the DCMS team extracted each sub-item from the original dataset (being more specific, but as a result, potentially removing them from surrounding context), these may have been coded differently, suggesting a limitation to this portion of our work.

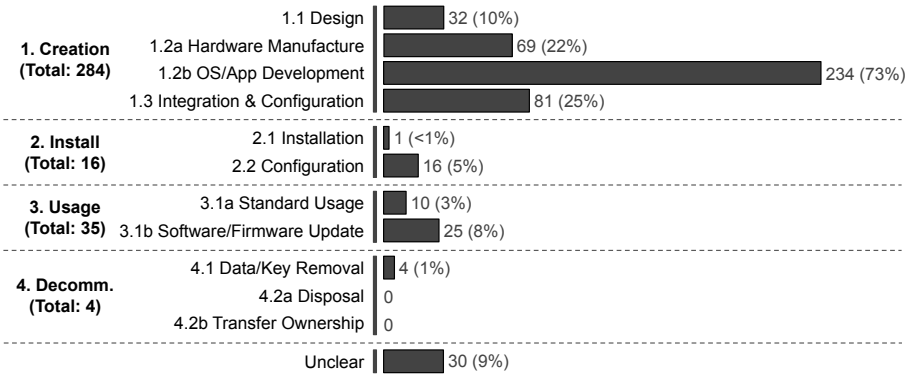


Fig. 7. Number of actionable practices suitable to implement at each lifecycle phase (Fig. 1). Total of all numbers and percentages exceed actionable practice total (320), major phase totals, and 100% as practices may be implemented in multiple phases. Percentages are proportion of 320 total actionable practices.

5.4 Advice Based on IoT Lifecycle

Fig. 7 highlights the results categorizing each phase in the IoT lifecycle (Fig. 1) and the number of advice items that are practices from the DCMS collection that could be carried out in each. All lifecycle phases associated with each practice (Section 4.2) were combined to produce the value of each bar. For example, if an item was coded as taking place in both the OS/App Development (1.2b) and Design (1.1) phases, +1 would be added to each of these counts in the figure.

The practice distribution among phases paints an important picture for the general execution of best practices: 89% (284/320) of practices have at least one possible phase (as they could take place in multiple phases) where they could be implemented while in the manufacturer's hands (i.e., the Creation phase), indicating that the designers and manufacturers are in a position to implement them. The OS/app developers (of Phase 1.2b, see Fig. 1) themselves have 73% (234/320) of practices targeted towards (or at least in a position to implement) them. This follows from many advice items being software-related, thus requiring implementation during the development phases where a large number of software development-focused stakeholders are involved versus when the product is in the end-user's hands. While this finding may seem self-evident, it draws focus to the importance of attention to security during this phase.

6 RELATED WORK

Formal definitions for best practice are rare in academic literature. King [41] was discussed in Section 3. Shostack et al. [60, pp.36–38] discuss modern uses of the term best practices. Tschofenig and Baccelli [65] discuss efforts by ENISA and the IETF to provide recommendations. They categorize technical and organizational areas to be considered for the secure development and use of IoT devices. Moore et al. [47] pursue best practices for IoT, specifically regarding network-based attacks. Based on our definitions herein, most of their advice items are not actionable (thereby not practices). Alrawi et al. [6] analyze and systematize work on home-based IoT security and propose a methodology for evaluating the security of home-based systems. They note that best practices are "readily available", but provide neither definitions nor specific references. Assal and Chiasson [7] note that technical detail for software development security practices is inconsistent, and report frequent non-compliance with common security advice. Redmiles et al. [56] analyze end-user security advice and find through a user-study that most advice is perceived as being actionable to the users, but it is unclear to them which of the actionable advice is the most important to follow (i.e., which advice to prioritize

from the set). Acar et al. [1] analyze software development security advice and find that software developers often find security advice to be inadequate for their needs, and lacking resources (e.g., implementation examples, tutorials) to help them understand the guidance.

A number of government and industrial agencies provide advice for IoT, for both manufacturers and groups looking to acquire IoT devices for their organizations. ENISA [25] published an expansive document about IoT security. This includes a substantial set of security recommendations, but also useful contextual and informative sections such as (to name a select few) the document's target audience (cf. Section 3.2), an overview of what IoT is and the relevant components, threat and risk analyses, and technical measures for implementation (which seem to be positioned as technical steps to complement other security advice). ETSI [23] (cf. Section 2.2) provides a series of baseline requirements for IoT security. These requirements use the 13 DCMS guidelines [20] as general topic headers (and include a new one of their own), but provide more detail about what technical steps need to be taken, leading to a more actionable set of IoT security advice. To supplement the baseline requirements document, ETSI provides a support document for how to confirm conformance with the advice therein, noting that it is independent from an assurance scheme [24]. Assurance is historically associated with products for governmental use [10, Chapters 18–21] [33, Chapter 13], but is typically too expensive or otherwise unsuitable to the consumer space. For DCMS documents [18, 20, 21] used herein, see Section 2.2. NIST published four documents (drafts at the time of this writing) surrounding the interaction between US federal government agencies and IoT manufacturers [50]. Three documents [27–29] assist IoT manufacturers to produce secure devices specifically for use in the US federal government by providing suggested technical and non-technical baseline guidance, and one [30] for government agencies to know what features or characteristics they should desire when procuring IoT devices.

RFC 8576 [32] proposes a generic lifecycle model of an IoT device, presented as a simplified model. NIST's SP 800-27 Rev. A [62] outlines five major lifecycle phases and suggests 33 IT security principles, categorized into these phases. While developed independently, our lifecycle (Section 2.1) has similarities, e.g., design/development, primary usage, and disposal/end-of-life phases. The NIST SP suggests that many principles are vital to positive security outcomes across many phases, not just with one single phase each (implying important principles for phases other than design). NIST SP 800-160 [57] outlines a taxonomy of 32 security design principles covering three areas of systems security: "security architecture and design", "security capability and intrinsic behaviors", and "lifecycle security"; the latter two being less specifically about the design phase.

Morgner et al. [48] explore the relationship between efforts in formal IoT technical standards and the (unfortunate) reality of the economics of IoT security and its implications for the general security of manufacturers.

7 DISCUSSION & CONCLUDING REMARKS

The basic concept of best practices is familiar to non-experts. Our analysis of a wide selection of security advice found conflation of the ideas of security goals (outcomes) and the steps or methods by which they may be reached (practices). We highlighted an important characteristic for security advice: whether or not it is actionable. We offer uniform, consistent terminology (Section 3) that characterizes and separates concepts. This allowed systematic exploration that began with generic discussion and analytic classification and was cross-checked through specific focus on consumer IoT devices and their lifecycle.

A main contribution was analysis of a large collection of IoT security advice. In particular, we examined how actionable current advice is, and what types of advice (i.e., the codes of Fig. 4) are being recommended. Our main focus has been a set of advice compiled and used by the DCMS, but originating from other organizations. In our

exploration of what types of advice are being recommended, iterative inductive coding was used to create a codebook that represents the dataset (Section 2.2). To objectively assign a code to each item in the dataset, we designed a coding tree. We suggest that IoT security advice-giving organizations consider using the coding tree and methodology of Section 4 to measure whether advice is actionable (among other more fine-grained classes), and take steps to improve the advice's actionability or target a desired code (Fig. 4). Conducting our analysis of 1013 advice items from industrial, governmental, and academic sources (Section 5), we were surprised to find that the majority are not explicit practices that can be followed, but non-actionable advice. Among the practices we identified as actionable, 73% are suitable to implement in the OS/App Development lifecycle phase of an IoT device (Section 5.4), thus by the product manufacturer and its software development partners. Since poor security practices early in the lifecycle accrue a *security debt*, with consequences in later phases (analogous to *tech debt* where technical shortcuts during development incur later costs [43]), this highlights the fundamental role of software developers to underpin security for aspects that they alone are in a position to control.

As the Internet of computers has grown into the Internet of things, an old problem remains: how to ensure that security best practices are followed. An open question is whether the research community can find ways to help advice-givers (including governments) to compile more effective guidance, and have manufacturers embrace it. Our work argues that currently, even in the security and technology communities (not to mention the general public), ambiguity surrounds the language of technical *best practices*—such that arguably, the term does as much damage to the security community as help. One theoretical path forward (to provoke thoughts, more than as a practical suggestion) is to seek agreement within the technical security community that the term itself is vague and nebulous, and its use should be boycotted.

We suggest that organizations proposing and endorsing "best practice" advice should have a clear idea of whether they are recommending practices, specifying baseline security requirements, or simply offering advice about good principles to think about. If the goal is that relevant stakeholders adopt and implement specific practices aiming to reduce security exposures, it is imperative that (actionable) best practices be identified and clearly stated, versus vague outcomes—lest the target stakeholders be unable to map advice to a concrete practice, even if so motivated. In summary: if security experts do not find guidelines clearly actionable, we should not expect (security non-expert) manufacturers to magically find a way to adopt and implement the advice. The economic motivation of manufacturers [48] (keeping in mind markets for lemons [2]), their poor track record in IoT security, and lack of accountability for vulnerabilities, point to a worrisome future. We hope that our work is a step towards improving the efficacy of advice on best practices.

REFERENCES

[1] Yasemin Acar, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L. Mazurek, and Sascha Fahl. 2017. Developers Need Support, Too: A Survey of Security Advice for Software Developers. *IEEE Cybersecurity Development (SecDev)* (2017), 22–26.

[2] George A. Akerlof. 1970. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics* 84, 3 (1970), 488–500.

[3] Alliance for Internet of Things Innovation (AIOTI). 2015. Report: Working Group 4—Policy. https://aioti.eu/wp-content/uploads/2017/03/AIOTIWG04Report2015-Policy-Issues.pdf

[4] Alliance for Internet of Things Innovation (AIOTI). 2016. AIOTI Digitisation of Industry Policy Recommendations. https://aioti.eu/wp-content/uploads/2017/03/AIOTI-Digitisation-of-Ind-policy-doc-Nov-2016.pdf

[5] Alliance for Internet of Things Innovation (AIOTI). 2016. Report on Workshop on Security and Privacy in the Hyper connected World. https://aioti-space.org/wp-content/uploads/2017/03/AIOTI-Workshop-on-Security-and-Privacy-in-the-Hyper-connected-World-Report-20160616_vFinal.pdf

[6] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *IEEE Symp. Security and Privacy*.

[7] Hala Assal and Sonia Chiasson. 2018. Security in the Software Development Lifecycle. In *Symp. on Usable Privacy and Security (SOUPS)*. USENIX, 281–296.

[8] AT&T. 2016. The CEO's Guide to Securing the Internet of Things. https://www.business.att.com/cybersecurity/docs/exploringiotsecurity.pdf

[9] Australian Cyber Security Centre. 2019. Securing the Internet of Things for Consumers (DRAFT). https://www.homeaffairs.gov.au/reports-and-pubs/files/code-of-practice.pdf

[10] Matt Bishop. 2003. *Computer Security: Art and Science*. Addison-Wesley.

[11] Harold Boley, Micha Meier, Chris Moss, Michael M Richter, and A. A. Voronkov. 1991. Declarative and Procedural Paradigms - Do They Really Compete?. In *International Workshop on Processing Declarative Knowledge*. Springer, 383–398.

[12] Broadband Internet Technical Advisory Group (BITAG). 2016. Internet of Things (IoT) Security and Privacy Recommendations. http://www.bitag.org/documents/BITAG_Report_-_Internet_of_Things_(IoT)_Security_and_Privacy_Recommendations.pdf

[13] CableLabs. 2017. A Vision for Secure IoT. https://www.cablelabs.com/insights/vision-secure-iot/

[14] City of New York (NYC) Guidelines for the Internet of Things. 2019. Privacy + Transparency. https://iot.cityofnewyork.us/privacy-and-transparency/

[15] City of New York (NYC) Guidelines for the Internet of Things. 2019. Security. https://iot.cityofnewyork.us/security/

[16] Cloud Security Alliance (CSA). 2015. Security Guidance for Early Adopters of the Internet of Things (IoT). https://downloads.cloudsecurityalliance.org/whitepapers/Security_Guidance_for_Early_Adopters_of_the_Internet_of_Things.pdf

[17] Cloud Security Alliance (CSA). 2016. Future-proofing the Connected World: 13 Steps to Developing Secure IoT. https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/future-proofing-the-connected-world.pdf

[18] Copper Horse Ltd. 2019. Mapping Security & Privacy in the Internet of Things. https://iotsecuritymapping.uk/

[19] George Corser, Glenn A. Fink, Mohammed Aledhari, Jared Bielby, Rajesh Nighot, Sukanya Mandal, Nagender Aneja, Chris Hrivnak, and Lucian Cristache. 2017. IoT Security Principles and Best Practices. https://internetinitiative.ieee.org/images/files/resources/white_papers/internet_of_things_feb2017.pdf

[20] Department for Digital, Culture, Media & Sport (DCMS). 2018. Code of Practice for Consumer IoT Security. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/773867/Code_of_Practice_for_Consumer_IoT_Security_October_2018.pdf

[21] Department for Digital, Culture, Media & Sport (DCMS). 2018. Mapping of IoT Security Recommendations, Guidance and Standards to the UK's Code of Practice for Consumer IoT Security. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/774438/Mapping_of_IoT__Security_Recommendations_Guidance_and_Standards_to_CoP_Oct_2018.pdf

[22] European Telecommunications Standards Institute (ETSI). 2019. CYBER; Cyber Security for Consumer Internet of Things. https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf

[23] European Telecommunications Standards Institute (ETSI). 2020. CYBER; Cyber Security for Consumer Internet of Things: Baseline Requirements. https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf

[24] European Telecommunications Standards Institute (ETSI). 2021. CYBER; Cyber Security for Consumer Internet of Things: Conformance Assessment of Baseline Requirements (DRAFT). https://docbox.etsi.org/CYBER/CYBER/Open/Latest_Drafts/CYBER-0050v009-TS103701-Cybersecurity-assessment-for-consumer-IoT-product.pdf Document version V0.0.9 (2021-05).

[25] European Union Agency for Cybersecurity (ENISA). 2017. Baseline Security Recommendations for IoT. https://www.ENISA.europa.eu/publications/baseline-security-recommendations-for-iot

[26] European Union Agency for Network and Information Security (ENISA). 2015. Security and Resilience of Smart Home Environments. https://www.ENISA.europa.eu/publications/security-resilience-good-practices

[27] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, and Rebecca Herold. 2020. Draft NISTIR 8259B—IoT Non-Technical Supporting Capability Core Baseline. NIST.

[28] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, and Rebecca Herold. 2020. Draft NISTIR 8259C—Creating a Profile Using the IoT Core Baseline and Non-Technical Baseline. NIST.

[29] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, and Rebecca Herold. 2020. Draft NISTIR 8259D—Profile Using the IoT Core Baseline and Non-Technical Baseline for the Federal Government. NIST.

[30] Michael Fagan, Jeffrey Marron, Kevin G. Brady Jr., Barbara B. Cuthill, Katerina N. Megas, and Rebecca Herold. 2020. Draft SP 800-213—IoT Device Cybersecurity Guidance for the Federal Government: Establishing IoT Device Cybersecurity Requirements. NIST.

[31] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. 2009. Declarative Versus Imperative Process Modeling Languages: The Issue of Understandability. In *Enterprise, Business-Process and Information Systems Modeling*. Springer, 353–366.

[32] Oscar Garcia-Morchon, Sandeep S. Kumar, and Mohit Sethi. 2019. RFC8576: Internet of Things (IoT) Security: State of the Art and Challenges.

[33] Dieter Gollmann. 2011. *Computer Security, 3rd Edition*. Wiley.

[34] Paul A. Grassi and 12 others. 2017. SP 800-63B—Digital Identity Guidelines: Authentication and Lifecycle Management. NIST.

[35] GSMA. 2017. IoT Security Guidelines for Endpoint Ecosystems—Version 2.0. https://www.gsma.com/iot/wp-content/uploads/2017/10/CLP.13-v2.0.pdf

[36] Nicholas Huaman, Sabrina Amft, Marten Oltrogge, Yasemin Acar, and Sascha Fahl. 2021. They Would do Better if They Worked Together: The Case of Interaction Problems Between Password Managers and Websites. In *IEEE Symp. Security and Privacy*. 1626—-1640.

[37] Wei Huang, Afshar Ganjali, Beom Heyn Kim, Sukwon Oh, and David Lie. 2015. The State of Public Infrastructure-as-a-Service Cloud Security. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 1–31.

[38] IoT Security Foundation. 2017. IoT Security Compliance Framework 1.1. https://www.iotsecurityfoundation.org/wp-content/uploads/2017/12/IoT-Security-Compliance-Framework_WG1_2017.pdf

[39] IoT Security Initiative. 2018. Security Design Best Practices. https://www.iotsi.org/security-best-practices

[40] Erica Johnson. 2020. Online Banking Agreements Protect Banks, Hold Customers Liable for Losses, Expert Says. *Canadian Broadcasting Corporation* (9 Feb 2020). https://www.cbc.ca/news/business/online-banking-agreements-1.5453192

[41] Guy King. 2000. Best Security Practices: An Overview. In *National Information Systems Security Conference*.

[42] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017), 80–84.

[43] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. 2012. Technical Debt: From Metaphor to Theory and Practice. *IEEE Software* 29 (2012), 18–21.

[44] Greg Lindsay, Beau Woods, and Joshua Corman. 2016. Smart Homes and the Internet of Things. https://www.atlanticcouncil.org/wp-content/uploads/2016/03/Smart_Homes_0317_web.pdf

[45] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact* 3, CSCW (Nov 2019), 1–23.

[46] Microsoft. 2018. Security best practices for Internet of Things (IoT). https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-best-practices

[47] Keith Moore, Richard Barnes, and Hannes Tschofenig. July, 2017. Best Current Practices (BCP) for IoT Devices. https://tools.ietf.org/html/draft-moore-iot-security-bcp-01

[48] Philipp Morgner and Zinaida Benenson. 2018. Exploring Security Economics in IoT Standardization Efforts. In *Workshop on Decentralized IoT Security and Standards (DISS)*.

[49] NIST. 2001. *Announcing the Advanced Encryption Standard (AES).* Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 197. U.S. Department of Commerce.

[50] NIST. 2020. NIST Releases Draft Guidance on Internet of Things Device Cybersecurity. https://www.nist.gov/news-events/news/2020/12/nist-releases-draft-guidance-internet-things-device-cybersecurity

[51] Online Trust Alliance (OTA). 2017. IoT Security & Privacy Trust Framework v2.5. https://www.internetsociety.org/wp-content/uploads/2018/05/iot_trust_framework2.5a_EN.pdf

[52] Open Web Application Security Project (OWASP). 2010. OWASP Secure Coding Practices Quick Reference Guide. https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf

[53] PlainTextOffenders.com. 2021. *About.* https://plaintextoffenders.com/about/

[54] Jon Postel, Yakov Rekhter, and Tony Li. 1995. *Best Current Practices.* RFC 1818. IETF.

[55] PSA Certified. 2019. Critical security questions for chip vendors, OS providers and OEMs. https://www.psacertified.org/app/uploads/2019/02/JSADEN001-PSA_Certified_Level_1-1.0Web.pdf

[56] Elissa M. Redmiles, Noel Warford, Amritha Jayanti, Aravind Koneru, Sean Kross, M. Morales, R. Stevens, and Michelle L. Mazurek. 2020. A Comprehensive Quality Evaluation of Security and Privacy Advice on the Web. In *USENIX Security Symposium*.

[57] Ron Ross, Michael McEvilley, and Janet Carrier Oren. 2016. SP.800-160—Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems. NIST.

[58] Jerome H. Saltzer and Michael D. Schroeder. 1975. The Protection of Information in Computer Systems. *Proc. IEEE* 63, 9 (1975), 1278–1308.

[59] Behcet Sarikaya, Mohit Sethi, and Dan Garcia-Carrillo. 2019. Internet Draft: Secure IoT Bootstrapping: A Survey. Document: draft-sarikaya-t2trg-sbootstrapping-05.

[60] Adam Shostack and Andrew Stewart. 2008. *The New School of Information Security.* Pearson Education.

[61] Abhay Soorya and 21 others. 2018. IoT Security Compliance Framework 2.0. https://www.iotsecurityfoundation.org/wp-content/uploads/2018/12/IoTSF-IoT-Security-Compliance-Framework-Release-2.0-December-2018.pdf

[62] Gary Stoneburner, Clark Hayden, and Alexis Feringa. 2004. SP 800-27 RevA—Engineering Principles for Information Technology Security (A Baseline for Achieving Security). NIST.

[63] Sven Schrecker and 14 others. 2016. Industrial Internet of Things Volume G4: Security Framework v1.0. https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB-3.pdf

[64] David R. Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation* 27, 2 (Jun 2006), 237–246.

[65] Hannes Tschofenig and Emmanuel Baccelli. 2019. Cyberphysical Security for the Masses: A Survey of the Internet Protocol Suite for Internet of Things Security. *IEEE Security & Privacy* 17, 5 (Sep 2019), 47–57.

[66] US National Telecommunications and Information Administration (NTIA). 2017. Voluntary Framework for Enhancing Update Process Security. https://www.ntia.doc.gov/files/ntia/publications/ntia_iot_capabilities_oct31.pdf

[67] US Senate. 2017. Bill—S.1691 - Internet of Things (IoT) Cybersecurity Improvement Act of 2017 (Bill). https://www.congress.gov/bill/115th-congress/senate-bill/1691/text?format=txt

[68] Felix Wortmann and Kristina Flüchter. 2015. Internet of Things. *Business & Information Systems Engineering* 57 (2015), 221–224.

A CODING TREE DETAILED ANNOTATION

Items Q1–Q11 (questions) and the adjacent text below serve as the detailed annotation for the coding tree (Fig. 5 discussed in Section 4). These annotations were used by coders if they were unsure about exactly what each question is asking, or if they needed further details to answer a particular question regarding an advice item they were coding.

Q1. Is the advice conveyed in unambiguous language, and relatively focused? Does the advice make sense from a language perspective (e.g., it is a sentence that you can read and makes sense), unambiguous (i.e., you can tell what they are trying to convey from a language perspective, not technical), and not multiple items grouped into one piece of advice? Is the advice focused on one topic, whether it is a step to take, an outcome to achieve, or security principle? If the advice seems to have multiple topics being discussed or has multiple outcomes it wants an implementer to reach, this would be considered unfocused.

Q2. Is it arguably helpful for security? Is the advice arguably useful for pursuing security in some way? Does it seem like it will help improve security outcomes rather than processes unrelated to security?

Q3. Is it focused more on a desired outcome than how to achieve it? Is the advice a high-level outcome rather than some method (or meta-outcome) for how to achieve an outcome? E.g., *data is secured in transit* would be an outcome because it is a desired goal or state, whereas *encrypt data in transit* is not because it explains a method for achieving that outcome (in this case, encryption). *Encryption* may be considered a meta-outcome, as it is not meaningful to the end-user's ultimate goal of protected data.

Q4. Does it suggest a security technique, mechanism, software tool, or specific rule? Is the item a method used in achieving/following the advice? E.g., *encryption* or *replacing a password with black dots* are techniques/mechanisms, but *secure data* or *making the password unreadable* are not. An example of a specific rule: *no hard-coded credentials*—this is a rule that is fairly specific as to its goal and would be followed like a practice, but not necessarily with actionable steps.

Q5. Does it describe or imply steps or explicit actions to take? Does the advice suggest actionable technical steps (one or more) that suffice to follow the advice? It has sufficient detail to suggest a step/action to take.

Actionable: Involving a known, unambiguous sequence of steps, whose means of execution is generally understood.

Q6. Is it viable to accomplish with reasonable resources? Could the advice item be followed with an acceptable cost?. E.g., the advice would not take years to follow, or have cost out of line with the anticipated benefit.

Q7. Is it intended that the end-user carry this advice out? Does the item suggest that the end-user will be responsible for carrying out this practice? Note that end-users first interact with devices after the Creation phase.

Q8. Is it intended that a security expert carry this item out? Does following this advice require an expert understanding of security and security implementation in order to properly follow the advice? Someone following this advice item would have to be an expert in security to be able to understand it and successfully follow it, or be capable of extracting actionable steps from an otherwise non-actionable item based on their experience.

Q9. Is it a general policy, general practice, or general procedure? Is the item a security policy (general rule) to improve security, but is not explicit about what technical means is used? These are less actionable (akin to incompletely specified practices—see definition in Q5), and are not technically explicit. A general policy often has more emphasis on what is (dis)allowed (or may be a general rule closely related to a desired outcome), rather than on how to achieve it.

Q10. Is it a broad approach or security property? Is the item a general way or general strategy, or a property that would improve security? A security property is a characteristic or attribute of a system related to security. E.g., an *open design*.

Q11. Does it relate to a principle in the design? Some principles relate to the core design phase of the product/system rather than later lifecycle phases.